

Regularization for Multiple Kernel Learning via Sum-Product Networks

Abstract

In this paper, we are interested in constructing general graph-based regularizers for multiple kernel learning (MKL) given a structure which is used to describe the way of combining basis kernels. Such structures are represented by sum-product networks (SPNs) in our method. Accordingly we propose a new convex regularization method for MKL based on a path-dependent kernel weighting function which encodes the entire SPN structure in our method. Under certain conditions and from the view of probability, this function can be considered to follow multinomial distributions over the weights associated with product nodes in SPNs. We also analyze the convexity of our regularizer and the complexity of our induced classifiers, and further propose an efficient wrapper algorithm to optimize our formulation. In our experiments, we apply our method to

1. Introduction

In real world, information can be always organized under certain structures, which can be considered as the prior knowledge about the information. For instance, to understand a 2D scene, we can decompose the scene as “scene \rightarrow objects \rightarrow parts \rightarrow regions \rightarrow pixels”, and reason the relations between them (Ladicky, 2011). Using such structures, we can answer questions like “what and where the objects are” (Ladicky et al., 2010) and “what the geometric relations between the objects are” (Desai et al., 2011). Therefore, information structures are very important and useful for information integration and reasoning.

Multiple kernel learning (MKL) is a powerful tool for information integration, which aims to learn optimal kernels for the tasks by combining different basis kernels linearly (Rakotomamonjy et al., 2008; Xu et al., 2010; Kloft et al., 2011) or nonlinearly (Bach, 2008; Cortes et al., 2009; Varma & Babu, 2009) with certain constraints on kernel weights. In (Gönen & Alpaydm, 2011) a nice review

on different MKL algorithms was given, and in (Tomioka & Suzuki, 2011) some regularization strategies on kernel weights were discussed.

Recently, structure induced regularization methods have been attracting more and more attention (Bach et al., 2011; Maurer & Pontil, 2012; van de Geer, 2013; Lin et al., 2014). For MKL, Bach (Bach, 2008) proposed a hierarchical kernel selection (or more precisely, kernel decomposition) method for MKL based on directed acyclic graph (DAG) using structured sparsity-induced norm such as ℓ_1 norm or block ℓ_1 norm (Jenatton et al., 2011). Szafranski et al. (Szafranski et al., 2010) proposed a composite kernel learning method based on tree structures, where the regularization term in their optimization formulation is a composite absolute penalty term (Zhao et al., 2009). Though the structure information of how to combine basis kernels are taken into account when constructing regularizers, however, the weights of nodes in the structures appear independently in these regularizers. This type of formulations actually weaken the connections between the nodes in the structures, making the learning rather easy.

To distinguish our work from previous research on regularization for MKL:

- (1) We utilize the sum-product networks (SPNs) (Poon & Domingos, 2011) to describe the procedure of combining basis kernels. An SPN is a more general and powerful deep graphical representation consisting of only sum nodes and product nodes. Considering that the optimal kernel in MKL is created using summations and/or multiplications between non-negative weights and basis kernels, this procedure can be naturally described by SPNs. Notice that in general SPNs may not describe kernel embedding directly (Zhuang et al., 2011; Strobl & Visweswaran, 2013). However, using Taylor series we still can approximate kernel embedding using SPNs.
- (2) We accordingly propose a convex regularization method based on a new path-dependent kernel weighting function, which encodes the entire structures of SPNs. This function can be considered to follow multinomial distributions, involving much stronger connections between the node weights.

We also analyze the convexity of our regularizer and the

Rademacher complexity of the induced MKL classifiers. Further we propose an efficient wrapper algorithm to solve our problem, where the weights are updated using gradient descent methods (Palomar & Eldar, 2010).

The rest of this paper is organized as follows. In Section 2, we explain how to describe the kernel combination procedure using SPNs and our path-dependent kernel weighting function based on SPNs. In Section 3, we provide the details of our regularization method, namely SPN-MKL, including the analysis of regularizer convexity, Rademacher complexity, and our optimization algorithm. We show our experimental results and comparisons among different methods on

2. Path-dependent Kernel Weighting Function

2.1. Sum-Product Networks

A sum-product network (SPN) is a rooted directed acyclic graph (DAG) whose internal nodes are sums and products (\oplus and \otimes) and leaf nodes are basis kernels (K_1, \dots, K_n) (Poon & Domingos, 2011).

Given an SPN for MKL, we denote a *path* from the root node to a leaf node (i.e. kernel) in the SPN as $\mathbf{m} \in \mathcal{M}$ where \mathcal{M} consists of all the paths, and a *product node* as $v \in \mathcal{V}$ where \mathcal{V} consists of all the nodes. Along each path \mathbf{m} , we call the sub-path between any pair of adjacent sum nodes or between a leaf node and its adjacent sum node a *layer*, and denote it as \mathbf{m}_l ($l \geq 1$) and the number of layers along \mathbf{m} as $N_{\mathbf{m}}$. We denote the number of product nodes in layer \mathbf{m}_l as $N_{\mathbf{m}_l}$. We also denote the weights associated with path \mathbf{m} and the weight of the n^{th} product node in its layer \mathbf{m}_l as $\beta_{\mathbf{m}}$ and $\beta_{m_l^n}$, respectively, and $\beta_{\mathbf{m}}$ is a vector consisting of all $\{\beta_{m_l^n}\}_{\forall m_l^n}$. There is no associated weight for any sum node in the SPN.

Fig. 1 gives an example of constructing an SPN for basis kernel combination by embedding atomic SPNs into each other. *Atomic SPNs* in our method are the SPNs with *single* layer. Given an SPN as shown at the bottom left in Fig. 1 and the node weights, we can easily calculate the optimal kernel as $\mathbf{K}_{opt} = \beta_8(\beta_1\mathbf{K}_1 + \beta_2\mathbf{K}_2) + \beta_9(\beta_3\mathbf{K}_3 + \beta_4\mathbf{K}_4) \circ (\beta_5\mathbf{K}_5 + \beta_6\mathbf{K}_6 + \beta_7\mathbf{K}_7)$, where \circ denotes the entry-wise product between two matrices. Moreover, we can rewrite \mathbf{K}_{opt} as $\mathbf{K}_{opt} = \beta_8(\beta_1\mathbf{K}_1 + \beta_2\mathbf{K}_2) + \beta_9 \prod_{i=3}^4 \prod_{j=5}^7 \beta_i \beta_j (\mathbf{K}_i \circ \mathbf{K}_j)$, whose combination procedure can be described using the SPN at the bottom right in Fig. 1. Here $\forall i, j, \mathbf{K}_i \circ \mathbf{K}_j$ is a *path-dependent kernel*. For instance, the corresponding kernel for the path denoted by the red edges at the bottom right figure is $\mathbf{K}_4 \circ \mathbf{K}_6$. In fact, such kernel combination procedures for MKL can be always represented using SPNs in similar ways as shown at the bottom right figure.

Traditionally, SPNs are considered as probabilistic models

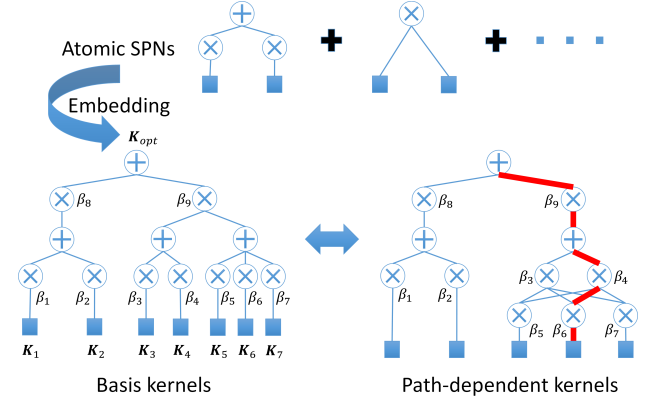


Figure 1. An example (i.e. bottom left) of constructing an SPN for basis kernel combination by embedding atomic SPNs into each other. All the weights (i.e. β 's) associated with product nodes in the SPN are learned in our method. The red edges in the bottom right graph denote a path from the root to a path-dependent kernel. This figure is best viewed in color.

and learned in an unsupervised manner (Gens & Domingos, 2012; Pecharz et al., 2013; Poon & Domingos, 2011). However, in our method we only utilize SPNs as representations to describe the kernel combination procedure, and learn the weights associated with their product nodes for MKL. In addition, from the aspect of structures for kernel combination, many existing MKL methods, e.g. (Rakotomamonjy et al., 2008; Cortes et al., 2009; Xu et al., 2010; Szafranski et al., 2010), can be considered as our special cases.

2.2. Our Kernel Weighting Function

Given an SPN and its associated weights β 's, we define our *path-dependent kernel weighting function* $g_{\mathbf{m}}(\beta_{\mathbf{m}})$ as:

$$\forall \mathbf{m} \in \mathcal{M}, g_{\mathbf{m}}(\beta_{\mathbf{m}}) = \prod_{l=1}^{N_{\mathbf{m}}} \prod_{n=1}^{N_{\mathbf{m}_l}} (\beta_{m_l^n})^{\frac{1}{N_{\mathbf{m}} N_{\mathbf{m}_l}}}. \quad (1)$$

Taking the red path in Fig. 1 for example, the kernel weighting function for this path is $g = \beta_9^{\frac{1}{2 \times 1}} \beta_4^{\frac{1}{2 \times 2}} \beta_6^{\frac{1}{2 \times 2}}$ with $N_{\mathbf{m}} = 2$, $N_{\mathbf{m}_1} = 1$, and $N_{\mathbf{m}_2} = 2$.

Given an SPN and $\forall \mathbf{m} \in \mathcal{M}$, suppose $\forall m_l^n, 0 \leq \beta_{m_l^n} \leq 1$. Then from the view of probability, since $\forall \mathbf{m} \in \mathcal{M}$, $N_{\mathbf{m}}$ and $N_{\mathbf{m}_l}$ are constants, $g_{\mathbf{m}}$ actually follows a multinomial distribution with variables $\beta_{\mathbf{m}}$ (ignoring the scaling factor). This is different from recent work (Gönen, 2012), where the kernel weights are assumed to follow multivariate normal distributions so that efficient inference can be performed. In contrast, our kernel weighting function is intuitively derived from the SPN structure, and under certain simple condition, it can guarantee the convexity of our

proposed regularizer (see our Lemma 1).

3. SPN-MKL

3.1. Formulation

Given $N_{\mathbf{x}}$ training samples $\{(\mathbf{x}_i, y_i)\}$, where $\forall i, \mathbf{x}_i \in \mathbb{R}^d$ is an input data vector and $y_i \in \{1, -1\}$ is its binary label, we formulate our SPN-MKL for binary classification as follows:

$$\min_{\mathcal{B}, \mathcal{W}, b} \sum_{\mathbf{m} \in \mathcal{M}} \left\{ \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{2 \cdot g_{\mathbf{m}}(\beta_{\mathbf{m}})} + \lambda \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{(\beta_{m_l^n})^{p_{m_l^n}}}{N_{\mathbf{m}} N_{\mathbf{m}_l}} \right\} + C \sum_i \ell(\mathbf{x}_i, y_i; \mathcal{W}, b) \quad (2)$$

$$\text{s.t. } \forall \beta \in \mathcal{B}, \beta \geq 0,$$

where $\mathcal{B} = \{\beta_{\mathbf{m}}\}_{\mathbf{m} \in \mathcal{M}}$ denotes the weight set, $\mathcal{W} = \{\mathbf{w}_{\mathbf{m}}\}_{\mathbf{m} \in \mathcal{M}}$ denotes the classifier parameter set, b denotes the bias term in the MKL classifier, $\lambda \geq 0$, $C \geq 0$, and $\mathcal{P} = \{p_v\}_{v \in \mathcal{V}}$ are predefined constants. Function $\forall i, \ell(\mathbf{x}_i, y_i; \mathcal{W}, b) = \max\{0, 1 - y_i [\sum_{\mathbf{m} \in \mathcal{M}} \mathbf{w}_{\mathbf{m}}^T \phi_{\mathbf{m}}(\mathbf{x}_i) + b]\}$ denotes the hinge loss function, where $\forall \mathbf{m} \in \mathcal{M}$, $\phi_{\mathbf{m}}(\cdot)$ denotes a path-dependent kernel mapping function and $(\cdot)^T$ denotes the matrix transpose operator, and our decision function for a given data $\bar{\mathbf{x}}$ is $f(\bar{\mathbf{x}}; \mathcal{B}, \mathcal{W}, b) = \sum_{\mathbf{m} \in \mathcal{M}} \mathbf{w}_{\mathbf{m}}^T \phi_{\mathbf{m}}(\bar{\mathbf{x}}) + b$. Moreover, we define $\forall \mathbf{m}, \forall l, \forall n, \lim_{\beta_{m_l^n} \rightarrow 0^+} \left\{ \|\mathbf{w}_{\mathbf{m}}\|_2^2 (\beta_{m_l^n})^{-\frac{1}{N_{\mathbf{m}} N_{\mathbf{m}_l}}} \right\} = 0$. This constraint guarantees the continuity of our objective function.

Note that unlike many existing MKL methods such as SimpleMKL (Rakotomamonjy et al., 2008), in Eq. 2 there is no ℓ_p norm constraint on the node weights β 's. This makes the weight learning procedure more flexible, only dependent on the data and the predefined SPN structure.

3.2. Analysis

In this section, we analyze the properties of our proposed regularizer and the Rademacher complexity of the induced MKL classifier.

Lemma 1. $\forall \mathbf{m} \in \mathcal{M}, f(\mathbf{w}_{\mathbf{m}}, \beta_{\mathbf{m}}) = \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{g_{\mathbf{m}}(\beta_{\mathbf{m}})}$ is convex over both $\mathbf{w}_{\mathbf{m}}$ and $\beta_{\mathbf{m}}$.

Proof. Clearly, f is continuous and differentiable with respect to $\mathbf{w}_{\mathbf{m}}$ and $\beta_{\mathbf{m}}$, respectively. Given arbitrary $\mathbf{w}_{\mathbf{m}}^{(0)}, \mathbf{w}_{\mathbf{m}}^{(1)}, \beta_{\mathbf{m}}^{(0)} \succeq \mathbf{0}$, and $\beta_{\mathbf{m}}^{(1)} \succeq \mathbf{0}$, where \succeq denotes the entry-wise \geq operator, based on the definition of a convex function, we need to prove $f(\mathbf{w}_{\mathbf{m}}^{(1)}, \beta_{\mathbf{m}}^{(1)}) \geq f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)}) + (\mathbf{w}_{\mathbf{m}}^{(1)} - \mathbf{w}_{\mathbf{m}}^{(0)})^T \frac{\partial f(\mathbf{w}_{\mathbf{m}}, \beta_{\mathbf{m}})}{\partial \mathbf{w}_{\mathbf{m}}} \Big|_{\mathbf{w}_{\mathbf{m}}=\mathbf{w}_{\mathbf{m}}^{(0)}} +$

$$(\beta_{\mathbf{m}}^{(1)} - \beta_{\mathbf{m}}^{(0)})^T \frac{\partial f(\mathbf{w}_{\mathbf{m}}, \beta_{\mathbf{m}})}{\partial \beta_{\mathbf{m}}} \Big|_{\beta_{\mathbf{m}}=\beta_{\mathbf{m}}^{(0)}}.$$

$$\therefore \frac{\partial f(\mathbf{w}_{\mathbf{m}}, \beta_{\mathbf{m}})}{\partial \mathbf{w}_{\mathbf{m}}} \Big|_{\mathbf{w}_{\mathbf{m}}=\mathbf{w}_{\mathbf{m}}^{(0)}} = \frac{2\mathbf{w}_{\mathbf{m}}^{(0)} f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)})}{\|\mathbf{w}_{\mathbf{m}}^{(0)}\|_2^2}, \text{ and}$$

$$\forall m_l^n, \frac{\partial f(\mathbf{w}_{\mathbf{m}}, \beta_{\mathbf{m}})}{\partial \beta_{m_l^n}} \Big|_{\beta_{m_l^n}=\beta_{m_l^n}^{(0)}} = -\frac{f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)})}{N_{\mathbf{m}} N_{\mathbf{m}_l} \cdot \beta_{m_l^n}^{(0)}},$$

\therefore By substituting above equations into our target, in the end we only need to prove that

$$\begin{aligned} f(\mathbf{w}_{\mathbf{m}}^{(1)}, \beta_{\mathbf{m}}^{(1)}) - \frac{2(\mathbf{w}_{\mathbf{m}}^{(1)})^T \mathbf{w}_{\mathbf{m}}^{(0)} f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)})}{\|\mathbf{w}_{\mathbf{m}}^{(0)}\|_2^2} \\ + f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)}) \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{1}{N_{\mathbf{m}} N_{\mathbf{m}_l}} \left(\frac{\beta_{m_l^n}^{(1)}}{\beta_{m_l^n}^{(0)}} \right) \\ \geq f(\mathbf{w}_{\mathbf{m}}^{(1)}, \beta_{\mathbf{m}}^{(1)}) - \frac{2(\mathbf{w}_{\mathbf{m}}^{(1)})^T \mathbf{w}_{\mathbf{m}}^{(0)} f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)})}{\|\mathbf{w}_{\mathbf{m}}^{(0)}\|_2^2} \\ + f(\mathbf{w}_{\mathbf{m}}^{(0)}, \beta_{\mathbf{m}}^{(0)}) \frac{g_{\mathbf{m}}(\beta_{\mathbf{m}}^{(1)})}{g_{\mathbf{m}}(\beta_{\mathbf{m}}^{(0)})} \\ \geq \left\| \frac{\mathbf{w}_{\mathbf{m}}^{(1)}}{\sqrt{g_{\mathbf{m}}(\beta_{\mathbf{m}}^{(1)})}} - \frac{\mathbf{w}_{\mathbf{m}}^{(0)} \sqrt{g_{\mathbf{m}}(\beta_{\mathbf{m}}^{(1)})}}{g_{\mathbf{m}}(\beta_{\mathbf{m}}^{(0)})} \right\|^2 \geq 0. \end{aligned} \quad (3)$$

Since Eq. 3 always holds, our lemma is proven. \square

Lemma 2. Given an SPN for MKL, $\forall \mathbf{m} \in \mathcal{M}, \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{g_{\mathbf{m}}(\beta_{\mathbf{m}})} \leq \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{1}{N_{\mathbf{m}} N_{\mathbf{m}_l}} \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{\beta_{m_l^n}}.$

Proof.

$$\frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{g_{\mathbf{m}}(\beta_{\mathbf{m}})} = \prod_{l,n} \left(\frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{\beta_{m_l^n}} \right)^{\frac{1}{N_{\mathbf{m}} N_{\mathbf{m}_l}}} \leq \sum_{l,n} \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{N_{\mathbf{m}} N_{\mathbf{m}_l} \beta_{m_l^n}}.$$

\square

From Lemma 1 and 2, we can see that our proposed regularizer is actually the lower bound of a family of widely used MKL regularizers (Rakotomamonjy et al., 2008; Xu et al., 2010; Gönen & Alpaydın, 2011; Kloft et al., 2011), involving much stronger connections between node weights.

Theorem 1 (Convex Regularization). *Our regularizer in Eq. 2 is convex if $\forall v \in \mathcal{V}, p_v \geq 1$.*

Proof. When $\forall v \in \mathcal{V}, p_v \geq 1, \forall m_l^n, \frac{(\beta_{m_l^n})^{p_{m_l^n}}}{N_{\mathbf{m}} N_{\mathbf{m}_l}}$ is convex over \mathcal{B} . Then based on Lemma 1, since the summation of convex functions is still convex, our regularizer is convex. \square

Theorem 2 (Rademacher Complexity). Denoting our MKL classifier learned from Eq. 2 as $f(\mathbf{x}; \mathcal{B}, \mathcal{W}, b) = \sum_{\mathbf{m}} \mathbf{w}_{\mathbf{m}}^T \phi_{\mathbf{m}}(\mathbf{x}) + b$ and our regularizer in Eq. 2 as

$$R(\mathcal{B}, \mathcal{W}; \lambda, \mathcal{P}) = R_1 + R_2 = \quad (4)$$

$$\sum_{\mathbf{m} \in \mathcal{M}} \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{2 \cdot g_{\mathbf{m}}(\beta_{\mathbf{m}})} + \lambda \sum_{\mathbf{m} \in \mathcal{M}} \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{(\beta_{m_l^n})^{p_{m_l^n}}}{N_{\mathbf{m}} N_{\mathbf{m}_l}},$$

the empirical Rademacher complexity of our classifier $\hat{R}(f)$ is upper-bounded by

$$\frac{2A}{N_{\mathbf{x}}} \cdot \min_{\mathcal{B}, \mathcal{W}, b} \left\{ R(\mathcal{B}, \mathcal{W}; 1, 1) + C \sum_i \ell(\mathbf{x}_i, y_i; \mathcal{W}, b) \right\}$$

where $N_{\mathbf{x}}$ denotes the total number of training samples, constant $A = \left(\sum_{i=1}^{N_{\mathbf{x}}} \sum_{\mathbf{m}} \mathbf{K}_{\mathbf{m}}(\mathbf{x}_i, \mathbf{x}_i) \right)^{\frac{1}{2}}$, and $\forall \mathbf{m}, \forall i, \mathbf{K}_{\mathbf{m}}(\mathbf{x}_i, \mathbf{x}_i) = \phi_{\mathbf{m}}(\mathbf{x}_i)^T \phi_{\mathbf{m}}(\mathbf{x}_i)$ denotes the i^{th} element along the diagonal of the path-dependent kernel matrix $\mathbf{K}_{\mathbf{m}}$.

Proof. Given the Rademacher variables σ 's, based on the definition of Rademacher complexity, we have

$$\begin{aligned} \hat{R}(f) &= \mathbb{E}_{\sigma} \left[\sup_{f \in \mathcal{F}(\mathcal{B}, \mathcal{W})} \left| \frac{2}{N_{\mathbf{x}}} \sum_{i=1}^{N_{\mathbf{x}}} \sigma_i f(\mathbf{x}_i; \mathcal{B}, \mathcal{W}, b) \right| \right] \\ &= \mathbb{E}_{\sigma} \left[\sup_{f \in \mathcal{F}(\mathcal{B}, \mathcal{W})} \left| \frac{2}{N_{\mathbf{x}}} \sum_{i=1}^{N_{\mathbf{x}}} \sigma_i \sum_{\mathbf{m} \in \mathcal{M}} \mathbf{w}_{\mathbf{m}}^T \phi_{\mathbf{m}}(\mathbf{x}_i) \right| \right] \\ &\leq \frac{4}{N_{\mathbf{x}}} \cdot \sup_{f \in \mathcal{F}} \left\{ \left[\sum_{\mathbf{m}} \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{2 \cdot g_{\mathbf{m}}(\beta_{\mathbf{m}})} \right]^{\frac{1}{2}} \cdot \left[\sum_{\mathbf{m}} g_{\mathbf{m}}(\beta_{\mathbf{m}}) \right]^{\frac{1}{2}} \right\} \\ &\quad \cdot \mathbb{E}_{\sigma} \left[\left\| \sum_{i=1}^{N_{\mathbf{x}}} \sum_{\mathbf{m}} \sigma_i \phi_{\mathbf{m}}(\mathbf{x}_i) \right\| \right] \\ &\leq \frac{2}{N_{\mathbf{x}}} \cdot \sup_{f \in \mathcal{F}} \left\{ \sum_{\mathbf{m}} \frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{2 \cdot g_{\mathbf{m}}(\beta_{\mathbf{m}})} + \sum_{\mathbf{m}} g_{\mathbf{m}}(\beta_{\mathbf{m}}) \right\} \cdot A \\ &\leq \frac{2A}{N_{\mathbf{x}}} \cdot \sup_{f \in \mathcal{F}} \left\{ \sum_{\mathbf{m}} \left[\frac{\|\mathbf{w}_{\mathbf{m}}\|_2^2}{2 \cdot g_{\mathbf{m}}(\beta_{\mathbf{m}})} + \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{\beta_{m_l^n}}{N_{\mathbf{m}} N_{\mathbf{m}_l}} \right] \right\} \\ &\leq \frac{2A}{N_{\mathbf{x}}} \cdot \min_{\mathcal{B}, \mathcal{W}, b} \left\{ R(\mathcal{B}, \mathcal{W}; 1, 1) + C \sum_i \ell(\mathbf{x}_i, y_i; \mathcal{W}, b) \right\} \end{aligned}$$

□

From Theorem 2 we can see that with $\lambda = 1$ and $\forall v \in \mathcal{V}, p_v = 1$, minimizing our objective function in Eq. 2 is equivalent to minimizing the upper bound of Rademacher complexity of our induced MKL classifier. To enhance the flexibility of our method, we allow λ and \mathcal{P} to be tuned according to datasets.

3.3. Optimization

To optimize Eq. 2, we adopt a similar learning strategy as used in (Xu et al., 2010) by updating the node weights \mathcal{B} and the classifier parameters (\mathcal{W}, b) alternatively.

3.3.1. LEARNING (\mathcal{W}, b) BY FIXING \mathcal{B}

We utilize the dual form of Eq. 2 to learn (\mathcal{W}, b) . Letting $\alpha \in \mathbb{R}^{N_{\mathbf{x}}}$ be the vector of Lagrange multipliers, and $\mathbf{y} \in \{-1, 1\}^{N_{\mathbf{x}}}$ be the vector of binary labels, then optimizing the dual of Eq. 2 is equivalent to maximizing the following problem:

$$\begin{aligned} \max_{\alpha} \quad & \mathbf{e}^T \alpha - \frac{1}{2} (\alpha \circ \mathbf{y})^T \left(\sum_{\mathbf{m} \in \mathcal{M}} g_{\mathbf{m}}(\beta_{\mathbf{m}}) \mathbf{K}_{\mathbf{m}} \right) (\alpha \circ \mathbf{y}) \\ \text{s.t.} \quad & \mathbf{0} \preceq \alpha \preceq C \mathbf{e}, \mathbf{y}^T \alpha = 0, \end{aligned} \quad (5)$$

where \preceq denotes the entry-wise \leq operator. Based on Eq. 5, the optimal kernel is constructed as $\mathbf{K}_{\text{opt}} = \sum_{\mathbf{m} \in \mathcal{M}} g_{\mathbf{m}}(\beta_{\mathbf{m}}) \mathbf{K}_{\mathbf{m}}$, and $\forall \mathbf{m} \in \mathcal{M}, \mathbf{w}_{\mathbf{m}} = g_{\mathbf{m}}(\beta_{\mathbf{m}}) \sum_i \alpha_i y_i \phi_{\mathbf{m}}(\mathbf{x}_i)$.

Therefore, the updating rule for $\|\mathbf{w}_{\mathbf{m}}\|_2^2$ is:

$$\forall \mathbf{m} \in \mathcal{M}, \|\mathbf{w}_{\mathbf{m}}\|_2^2 = g_{\mathbf{m}}(\beta_{\mathbf{m}})^2 (\alpha \circ \mathbf{y})^T \mathbf{K}_{\mathbf{m}} (\alpha \circ \mathbf{y}). \quad (6)$$

3.3.2. LEARNING \mathcal{B} BY FIXING (\mathcal{W}, b)

At this stage, minimizing our objective function in Eq. 2 is equivalent to minimizing $R(\mathcal{B}, \mathcal{W}; \lambda, \mathcal{P})$ in Eq. 4, provided that $\forall \beta \in \mathcal{B}, \beta \geq 1$. For further usage, we rewrite R_2 in Eq. 4 as follows:

$$R_2 = \sum_{v \in \mathcal{V}} \left(\sum_{\mathbf{m} \in \mathcal{M}(v)} \sum_{l=1}^{N_{\mathbf{m}}} \sum_{n=1}^{N_{\mathbf{m}_l}} \frac{\lambda}{N_{\mathbf{m}} N_{\mathbf{m}_l}} \right) \beta_v^{p_v}, \quad (7)$$

where $\mathcal{M}(v)$ denotes all the paths which pass through product node v .

Due to the complex structures of SPNs, in general there may not exist close forms to update \mathcal{B} . Therefore, we utilize gradient descent methods to update \mathcal{B} .

(i) **Convex Regularization with** $\forall v \in \mathcal{V}, p_v \geq 1$

Since in this case our objective function is already convex, we can calculate its gradient directly and use the following rule to update \mathcal{B} : $\forall v \in \mathcal{V}$,

$$\beta_v^{(k+1)} = \left[\beta_v^{(k)} - \eta_{k+1} \left(\nabla_{\beta_v} R_1(\mathcal{B}^{(k)}) + \nabla_{\beta_v} R_2(\mathcal{B}^{(k)}) \right) \right]_+ \quad (8)$$

where ∇_{β_v} denotes the first-order derivative operation over variable β_v , $(\cdot)^{(k)}$ denotes the value at the k^{th} iteration, $\eta_{k+1} \geq 0$ denotes the step size at the $(k+1)^{\text{th}}$ iteration, and $[\cdot]_+ = \max\{0, \cdot\}$.

Algorithm 1 SPN-MKL learning algorithm

Input : $\{(\mathbf{x}_i, y_i)\}_{i=1, \dots, N_x}$, an SPN, $\{p_v > 0\}_{\forall v \in \mathcal{V}}$,
 $\{\mathbf{K}_m\}_{\forall m \in \mathcal{M}}, C$
Output: $\alpha, \mathcal{B} = \{\beta_v\}_{\forall v \in \mathcal{V}}$
 Initialize the kernel weights so that $\forall v \in \mathcal{V}, \beta_v \geq 0$;
repeat
 Update α using Eq. 5 while fixing \mathcal{B} ;
 (For multiclass cases, update $\{\alpha_c\}_{c \in \mathcal{C}}$ using Eq. 10 while fixing \mathcal{B});
 Update $\forall m, \|\mathbf{w}_m\|_2^2$ using Eq. 6 while fixing α and \mathcal{B} ;
 (For multiclass cases, update $\forall m, \|\mathbf{w}_m\|_2^2$ using Eq. 11 while fixing $\{\alpha_c\}_{c \in \mathcal{C}}$ and \mathcal{B});
 foreach $v \in \mathcal{V}$ **do**
 if $p_v \geq 1$ **then**
 Update β_v using Eq. 8 while fixing \mathbf{w} ;
 else
 repeat
 Update $\beta_v^{(k+1)}$ using Eq. 8 and Eq. 9 while fixing \mathbf{w} ;
 until Converge;
 end
 end
until Converge;
return α, \mathcal{B}

(ii) Non-Convex Regularization with $\exists v \in \mathcal{V}, 0 < p_v < 1$

In this case, since our objective function can be decomposed into summation of convex (*i.e.* in R_2 all terms with $p_v \geq 1$) and concave (*i.e.* in R_2 all terms with $0 < p_v < 1$) functions, we can utilize Concave-Convex procedure (CCCP) (Yuille & Rangarajan, 2003) to optimize it. Therefore, the weight updating rule for nodes with $0 < p_v < 1$ is changed to:

$$\beta_v^{(k+1)} = \arg \min_{\beta_v \geq 0} \left\{ R_1 + \beta_v \nabla_{\beta_v} R_2(\mathcal{B}^{(k)}) \right\}. \quad (9)$$

Again Eq. 8 can be reused to solve Eq. 9 iteratively.

To summarize, as long as $\forall v \in \mathcal{V}, p_v > 0$, we can always optimize our objective function. We show our learning algorithm for binary SPN-MKL in Alg. 1. Note that once the weight of any product node is equal to 0, it will always keep zero, which indicates that the product node and all the paths that go through it can be deleted from the SPN permanently. This property can be used to simplify the SPN structure and accelerate the learning speed of our SPN-MKL.

3.4. Multiclass SPN-MKL

For multiclass tasks, we generate a single optimal kernel for all the classes, and correspondingly modify Eq. 5 and Eq. 6 for binary SPN-MKL without changing other steps. Using the “one vs. the-rest” strategy, the modification is

shown as follows:

$$\max_{\{\alpha_c\}_{c \in \mathcal{C}}} \sum_{c \in \mathcal{C}} \mathbf{e}^T \alpha_c \quad (10)$$

$$-\frac{1}{2}(\alpha_c \circ \mathbf{y}_c)^T \left(\sum_{m \in \mathcal{M}} g_m(\beta_m) \mathbf{K}_m \right) (\alpha_c \circ \mathbf{y}_c)$$

$$\text{s.t. } \forall c \in \mathcal{C}, \mathbf{0} \preceq \alpha_c \preceq C \mathbf{e}, \mathbf{y}_c^T \alpha_c = 0,$$

$$\forall m, \|\mathbf{w}_m\|_2^2 = g_m(\beta_m)^2 \sum_{c \in \mathcal{C}} (\alpha_c \circ \mathbf{y}_c)^T \mathbf{K}_m (\alpha_c \circ \mathbf{y}_c), \quad (11)$$

where $c \in \mathcal{C}$ denotes a class label c in a label set \mathcal{C} , α_c denotes a class-specific Lagrange multipliers, and \mathbf{y}_c denotes a binary label vector: if $\forall i, y_i = c$, then the i^{th} entry in \mathbf{y}_c is set to 1, otherwise, 0.

The learning algorithm for multiclass SPN-MKL is listed in Alg. 1 as well.

4. Experiments
References

- Bach, Francis. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, pp. 105–112, 2008.
- Bach, Francis, Jenatton, Rodolphe, Mairal, Julien, and Obozinski, Guillaume. Structured sparsity through convex optimization. *CoRR*, abs/1109.2397, 2011.
- Cortes, Corinna, Mohri, Mehryar, and Rostamizadeh, Afshin. Learning non-linear combinations of kernels. In *NIPS*, pp. 396–404, 2009.
- Desai, Chaitanya, Ramanan, Deva, and Fowlkes, Charles. Discriminative models for multi-class object layout. *IJCV*, 2011.
- Gens, Robert and Domingos, Pedro. Discriminative learning of sum-product networks. In Bartlett, P., Pereira, F.C.N., Burges, C.J.C., Bottou, L., and Weinberger, K.Q. (eds.), *NIPS*, pp. 3248–3256. 2012.
- Gönen, Mehmet. Bayesian efficient multiple kernel learning. In *ICML*, 2012.
- Gönen, Mehmet and Alpaydın, Ethem. Multiple kernel learning algorithms. *JMLR*, 12(July):2211–2268, 2011.
- Jenatton, Rodolphe, Audibert, Jean-Yves, and Bach, Francis. Structured variable selection with sparsity-inducing norms. *J. Mach. Learn. Res.*, 12:2777–2824, November 2011. ISSN 1532-4435.

- Kloft, Marius, Brefeld, Ulf, Sonnenburg, Sren, and Zien, Alexander. ℓ_p -norm multiple kernel learning. *JMLR*, 12: 953–997, 2011.
- Ladicky, Lubor. *Global Structured Models towards Scene Understanding*. PhD thesis, Oxford Brookes University, April 2011.
- Ladicky, Lubor, Russell, Chris, Kohli, Pushmeet, and Torr, Philip H. S. Graph cut based inference with co-occurrence statistics. In *ECCV’10*, pp. 239–253, 2010.
- Lin, Lijing, Higham, Nicholas J., and Pan, Jianxin. Covariance structure regularization via entropy loss function. *Computational Statistics & Data Analysis*, 72:315–327, 2014.
- Maurer, Andreas and Pontil, Massimiliano. Structured sparsity and generalization. *J. Mach. Learn. Res.*, 13: 671–690, March 2012. ISSN 1532-4435.
- Palomar, Daniel P. and Eldar, Yonina C. (eds.). *Convex optimization in signal processing and communications*. Cambridge University Press, Cambridge, UK, New York, 2010. ISBN 978-0-521-76222-9.
- Peharz, Robert, Geiger, Bernhard, and Pernkopf, Franz. Greedy part-wise learning of sum-product networks. volume 8189, pp. 612–627. Springer Berlin Heidelberg, 2013.
- Poon, Hoifung and Domingos, Pedro. Sum-product networks: A new deep architecture. In *UAI*, pp. 337–346, 2011.
- Rakotomamonjy, Alain, Rouen, Universit De, Bach, Francis, Canu, Sthpane, and Grandvalet, Yves. SimpleMKL. *JMLR* 9, pp. 2491–2521, 2008.
- Strobl, Eric and Visweswaran, Shyam. Deep multiple kernel learning. *CoRR*, abs/1310.3101, 2013.
- Szafranski, Marie, Grandvalet, Yves, and Rakotomamonjy, Alain. Composite kernel learning. *Mach. Learn.*, 79(1-2):73–103, May 2010. ISSN 0885-6125.
- Tomioka, Ryota and Suzuki, Taiji. Regularization strategies and empirical bayesian learning for MKL. *JMLR*, 2011.
- van de Geer, Sara. Weakly decomposable regularization penalties and structured sparsity. *Scandinavian Journal of Statistics*, 2013. To appear.
- Varma, Manik and Babu, Bodla Rakesh. More generality in efficient multiple kernel learning. In *ICML*, pp. 134, 2009.
- Xu, Zenglin, Jin, Rong, Yang, Haiqin, King, Irwin, and Lyu, Michael R. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pp. 1175–1182, 2010.
- Yuille, A. L. and Rangarajan, Anand. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, April 2003. ISSN 0899-7667.
- Zhao, Peng, Rocha, Guilherme, and Yu, Bin. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 2009.
- Zhuang, Jinfeng, Tsang, Ivor W., and Hoi, Steven C. H. Two-layer multiple kernel learning. In *AISTATS*, pp. 909–917, 2011.